Calhoun: The NPS Institutional Archive

Theses and Dissertations | Thesis Collection

1992-09

NPSNET: modeling the in-flight and terminal properties of ballistic munitions

Nash, David Allan

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/38574

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A257 602

DTIC
ELECTE
DEC 0 1 1992
S B D

THESIS

NPSNET: Modeling the In-Flight and Terminal Properties of Ballistic Munitions

by

David A. Nash

September 1992

Thesis Advisor: Dr. Michael J. Zyda
Co-Advisor David R. Pratt

92-30368

Approved for public release
distribution unlimited.

92 11 30 017

BEST
AVAILABLE COPY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
NPSNET: Modeling the In-Flight and Terminal Properties of Ballistic Munitions

12. PERSONAL AUTHOR(S)
NASH, David A.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	03/91 09/92	1992, September 10	61

16. SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Artificial intelligence, Artificial reality, Artillery, Autonomous agents, Ballistics, Computer graphics, Graphics, Modeling, Object oriented, Particle systems, Real-time, Simulation, Trajectory, Virtual reality, Virtual world

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Graphical computer simulations provide a means through which weapon prototyping and tactical evaluations can be conducted at low cost, without the risks associated with the movement of equipment and firing of weapons. Because of the widespread use of ballistic munitions in the armed forces, a fundamental aspect of the implementation of such military simulations is a physical model that governs ballistic behavior. The modified point-mass trajectory model is used to implement ballistic trajectories within NPSNET, a real-time, graphical, three-dimensional simulation. A parallel algorithm is used to simulate the visual characteristics of shrapnel-producing explosions.

A special case of ballistic trajectories involves the application of indirect fires. When a projectile travels along a curved path to the target area, rather than being propelled directly along the line of sight, much greater ranges can be achieved. This makes it possible to fire upon an enemy without directly exposing the firing elements to harm. As a result of these increased ranges, it is generally not possible for the firing element to acquire its own targets. Thus, an additional player is required to represent this tactic in a virtual world: the forward observer. An expert system is presented that mimics the cognitive contributions of a human forward observer.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Michael J. Zyda	22b. TELEPHONE (Include Area Code): (408) 646-2476; 22c. OFFICE SYMBOL: CS/37X

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited

NPSNET: Modeling the In-Flight and Terminal Properties of Ballistic Munitions

by

David Allan Nash
Captain, United States Army
B. S. Engineering, United States Military Academy, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1992

Author: [redacted]
David Allan Nash

Approved By: [redacted]
Michael J. Zyda, Thesis Advisor

[redacted]
David R. Pratt, Co-Advisor

[redacted]
Robert B. McGhee, Chairman,
Department of Computer Science

ABSTRACT

Graphical computer simulations provide a means through which weapon prototyping and tactical evaluations can be conducted at low cost, without the risks associated with the movement of equipment and firing of weapons. Because of the widespread use of ballistic munitions in the armed forces, a fundamental aspect of the implementation of such military simulations is a physical model that governs ballistic behavior. The modified point-mass trajectory model is used to implement ballistic trajectories within NPSNET, a real-time, graphical, three-dimensional simulation. A parallel algorithm is used to simulate the visual characteristics of shrapnel-producing explosions.

A special case of ballistic trajectories involves the application of indirect fires. When a projectile travels along a curved path to the target area, rather than being propelled directly along the line of sight, much greater ranges can be achieved. This makes it possible to fire upon an enemy without directly exposing the firing elements to harm. As a result of these increased ranges, it is generally not possible for the firing element to acquire its own targets. Thus, an additional player is required to represent this tactic in a virtual world: the forward observer. An expert system is presented that mimics the cognitive contributions of a human forward observer.

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Thanks are due to a host of persons who helped with this research. I greatly appreciate Robert Lieske and Jon Miller of Ballistics Research Laboratory for their cogent explanations of the modified point-mass trajectory model. The implementation of Mr. Lieske's ideas in a virtual world environment would not have been possible without his illuminating assistance. David Pratt provided a constant stream of resources, references, and advice, for which I am very grateful. Dr. Michael Zyda gave very valuable insight into his ideas about autonomous agents in virtual worlds. Dr. Yutaka Kanayama graciously unraveled a thorny issue of vector analysis for me, which allowed simplification of the solution algorithm for the ballistic equations of motion. CPT Jon Walter and CPT Pat Warren saved countless hours of debugging by catching coding errors.

My greatest thanks are for Tammy and Leigh for their patient and loving support during the preparation of this thesis.

I. INTRODUCTION

A. BACKGROUND

Public pressure to reduce expenditures for national defense, coupled with a desire to field state of the art defense technology in a timely manner have led the Department of Defense to examine ways to "short-circuit" the traditionally time-consuming process of designing and fielding new weapon systems (Atwood, 1991). The ideal technique would allow for rapid prototyping and testing, and permit changes to original specifications without need of an extensive re-engineering effort.

B. SYNTHETIC ENVIRONMENTS

1. General.

A rapidly maturing technology that could potentially produce significant savings in the pursuit of low-cost weapons prototyping is the development of graphical computer simulations. Variously known in the literature as "artificial reality", "virtual worlds", "synthetic environments", and "virtual reality", the science of simulation has advanced dramatically since the first purely statistical combat models. The ability to create an electronically manipulable milieu inside which visible representations of objects can be made to imitate the real-world objects that they resemble has achieved some notoriety in the lay press of late, but to be sure, some of what sounded like shades of Flash Gordon as few as five years ago has come to pass.

2. Flight simulators.

Graphical simulations have already seen extensive application in some areas of the military genre. The earliest examples of these included aircraft flight simulators used to train pilots prior to, or even in lieu of, actual aircraft use. A mock-up of an aircraft cockpit was slaved to a computer that presented the operator with a display that changed the viewpoint and orientation according to inputs received from the controls. Updating the display at a sufficiently rapid rate produced the illusion of motion through a three-

dimensional space according to the rules of flight. This first use of graphical simulations remains a prevailing use today, albeit in a much more sophisticated form. Zyda *et al* have developed an inexpensive simulator for the purpose of modeling the flight of a guided munition, and testing the application of this weapon to different tactical scenarios (Zyda, McGhee, Smith, and Streyle, 1987, pp. 10-11).

3. Two-dimensional ground combat models.

The first uses of graphical simulations for the purpose of modeling the actions of ground forces (many of which remain in use) used the computer display as a two-dimensional automated wargaming platform. The computer accepted input in textual form or from a mouse that corresponded to actions in the simulation. The display was used mainly to keep track of units' positions from an overhead (or "god's eye") view of the simulated terrain. Engagements between ground units were modeled by some stochastic means based on the empirical characteristics of the weapons involved. Examples of these kinds of simulators include ARTBASS (Department of the Army, 1987), and JANUS(A) (Department of the Army, 1986).

A disadvantage of these two-dimensional programs is that they offer a very large-scale view of the virtual world at hand. The real world is of course decidedly three-dimensional. A single user of such a system therefore is typically not presented with sufficient stimuli to persuade him that he is to any extent participating in the simulation that the computer is modeling. On the other hand, when the objective of the simulation is to train high-level staffs, the two-dimensional representation is a very familiar one, and thus completely satisfactory.

4. Three-dimensional models.

State of the art military graphical simulations are real-time, three-dimensional, interactive systems. They include the capability to visualize world objects such as terrain, vehicles, buildings, and vegetation in depth. The user can move over the terrain and look around. As the user's position and orientation changes, the display device changes

accordingly, presenting a view that corresponds approximately to the scene the user would obtain when similarly oriented in an authentic environment of the same construction. Since the display is updated in real time, the user has the sensation of actually experiencing the events that are being simulated. NPSNET (Zyda & Pratt, 1992), SIMNET (Pope, 1989), and JANUS-3D (Walter & Warren, 1992) are all simulators of this category.

The three-dimensional nature of the latter category of simulations gives them a powerful advantage over their two-dimensional counterparts. The 3D simulation, being the more general, has the greatest efficiency in terms of utilization of the optical bandwidth. There are many cues within a 3D context that are accessible for encoding information about the simulated environment that are unavailable or ludicrous in 2D. For example, shadows can offer hints to an observer about an object's spatial orientation (Charniak & McDermott, 1985, pp. 129-130). The increased capacity of information transmission allows for the representation of more complex objects, and in greater physical detail. This level of specificity is essential to the creation of believable synthetic environments.

5. Benefits.

The versatility of the typical 3D simulation makes it especially well-suited for a variety of cost-saving military applications. Flight simulations are a classic example. In a networked system, the simultaneous interaction of large numbers of individuals is possible. The availability of world-wide telecommunications networks makes it viable for units on one side of the globe to engage in mock combat against units on the other side. Through use of autonomous forces, troops may train for missions involving combat with an enemy without need for a human opposing force's participation.

Aside from the obvious desirability of enhanced cost-effectiveness, a far more attractive characteristic emerges from the consideration of graphical simulations: improved safety. The act of training human beings to conduct armed warfare is inherently very dangerous. It invariably requires frequent interaction at close quarters with explosives, heavy equipment, and petroleum products. Training of any significant duration produces

fatigue, which can lead to decreased motor capacity and poor judgement. Exposure to the weather may further degrade a fighting man's ability to take actions to protect himself against unforeseen hazards. If training can be conducted in a virtual setting, these threats to life and limb can be eliminated. Individuals can be trained in the operation or maintenance of various types of equipment without risk to themselves or the machinery. Mortal combat can be simulated without actual bloodshed, either accidental or intended. The 3D interactive virtual world offers the greatest hope among diverse technologies for maintaining a credible defensive capability in the face of waning public support for adequate funding.

C. AGENTS OF FORCE IN A VIRTUAL WORLD

In most simulations, it is useful to monitor the participant's status, and render a report at certain intervals with some kind of comparative analysis. In game-type simulations, this is known as "keeping score". For the military force-on-force analogue, a means is needed to determine the unit or individual that will prevail in the course of an engagement. In real combat, of course, the force that is able to inflict the greatest amount of damage upon its opponent is the victor. Thus, any simulation that models the interaction of military units necessarily implies the implementation of various kinds of weapons, since they are the principal means by which deadly force is applied.

The agents of force presently in use can be categorized according to the characteristics of their flight through the air. Ballistic munitions travel along a roughly parabolic trajectory, with only naturally occurring forces such as gravity, air friction, and wind determining their course after launch. Guided and rocket-propelled munitions rely upon the action of man-made forces such as the movement of flight control surfaces, or the forces generated by the expulsion of gases from a propellant source during the course of their flight to determine their point of impact. Both kinds of devices have found widespread use in the armed forces of the world. Weapons of the ballistic variety, however, are by far the most prevalent among ground forces. Main battle tanks, infantry fighting vehicles, most

field artillery, mortars, crew-served machineguns, and of course, the infantryman's rifle are all examples of ballistic weapons.

The focus of this thesis is an implementation of a ballistic weapons model within the context of a synthetic environment. Techniques appropriate for the simulation of in-flight and terminal characteristics are developed, as well as support modules necessary for the special case of indirect ballistic systems. The most significant characteristic of these techniques are that they operate in real-time, providing feedback to the event control loop with sufficient rapidity to allow the perception of cause and effect necessary to maintain the persuasiveness of the simulation.

II. NPSNET AND OTHER PREVIOUS WORK

A. NPSNET

NPSNET is a real-time, graphical simulation. Its focus is the interaction of ground and air vehicles in a combat environment (Zyda & Pratt, 1992). Any number of humans can participate through use of networked graphics workstations. At present, NPSNET provides for a company of autonomous vehicles that can be assigned missions in the manner of aggressor forces (Culpepper, 1992). By default, the remainder of the vehicles that exist in this virtual world behave in an autonomous fashion, albeit statically. These vehicles are assigned random velocities and orientations upon program start-up, and move over the simulated terrain accordingly. If fired upon, they will alternatively return fire, or retire at high speed. The capability exists to record and playback sequences of interaction, and to view these sequences from any viewpoint available in the world space.

The primary means by which deadly force is applied in NPSNET is through direct, line-of-sight fires. When a player sees an enemy vehicle on the screen, he visually orients himself so as to be able to fire in the direction of the target. A projectile object is generated at the moment of firing which follows a trajectory that is linear by default. Collision detection of the projectile is performed along its path, and a vehicle is "killed" if the projectile passes within some specified distance.

NPSNET provides an excellent platform upon which to implement realistic models of ballistic weapons. The availability of visual representations of a plethora of vehicles and objects sets the stage for a virtual firing range. Numerical models of the flight of various kinds of projectiles can be constructed, and their efficacy tested by direct observation within the simulator. The behavior of entities intended to control the application of direct and indirect fires may be examined visually as well.

B. SURVEY OF PREVIOUS WORK

The computation of ballistic trajectories using digital computers has a long and distinguished history. The venerable ENIAC was designed specifically for the purpose of

automating the tedious process of calculating weapons data for the United States Army (Tanenbaum, 1990, p. 16). Attempts to model ballistic weapons within the context of a real-time graphical simulation, however, have begun fairly recently.

1. Graphical Simulators.

a. SIMNET

SIMNET (Garvey & Monday, 1988) is a sophisticated three-dimensional military simulator. It makes use of specially designed hardware and software to provide a networked interactive virtual world. The flight of ballistic munitions are modeled using a modified static linear algorithm. Terminal visual effects are obtained by the intermittent presentation of texture maps. Kills are determined by a stochastic selection, using historical distributions of targeting probable errors. Direct fires are controlled by either of the following methods:

- by console operators playing the role of a vehicle commander,
- by Semi-Autonomous Forces (SAF) entities,
- by human SAF commanders.

Indirect fires in SIMNET must be controlled by the SAF commander, a human console operator that supervises the interaction of SAF forces with forces controlled by other humans using the system.

b. NPSNET

In previous versions of NPSNET, the paths of ballistic projectiles were calculated by direct interpolation between the launch position and the point of impact. Thus, indirect fires were not modeled. NPSNET also makes use of an animated sequence of texture maps to give a visual representation of terminal weapons effects. Kills are determined by performing 3D collision detection within the boundaries of two invisible concentric polygons surrounding the projectile. Objects detected as being within the outermost polygon are assessed damage; objects within the inner polygon are destroyed (Osborne, 1991). Monahan has developed a technique by which the kinematics of particles

generated in the decomposition of an exploding target can be modeled based on the force environment in which the explosion takes place (Monahan, 1991). This technique proved to be too computationally intensive to implement within NPSNET, given the state of the art in graphics hardware. Branley and Culpepper devised the means to create instances of platoon-sized elements in NPSNET which could react autonomously to the presence of other entities in the virtual world according to previously established mission instructions (Branley, 1992) (Culpepper, 1992).

c. FOST

The Forward Observer Simulation Trainer (FOST) was created by Drummond and Nizolak (Drummond and Nizolak, 1989). It is a graphical application that simulates the interface of the Digital Message Device, which is the means by which Army indirect fire spotters communicate with the firing elements. Since the primary intent of their work was to develop an automated means for soldiers to train with this particular interface device, the representation of the projectile's actual trajectory was not developed; rather, the projectiles appeared at the location specified by the observer at the appropriate time.

2. Particle systems of ballistic entities.

Reeves used a particle-based approach to model the terminal visual effects of an exploding "wall of fire" (Reeves, 1983, p. 365). This was a successful implementation using the abstraction of a dynamic particle system for a graphical simulation. Later, he extended the technique to include objects that reflect light, as well as emit it (Reeves and Blau, 1985 p. 313). More recently, Loke et al used a similar technique for representing the visual characteristics of fireworks (Loke, Tan, Seah, and Er, 1992, p. 33).

III. EXTERIOR BALLISTICS

A. DEFINITION

The study of *exterior ballistics* pertains to the description of the behavior of a ballistic projectile as it flies through the air. More specifically, it is concerned with the period of time between the moment the projectile leaves the barrel of the weapon from which it was fired, and the moment that it explodes in the air or strikes the ground. In a virtual world, being able to establish the position of a projectile as a function of time is fundamental to the successful modeling of any sort of weapon system. This capability is necessary in order to render a scene that contains the projectile, as well as to perform collision detection with the other objects in the world. In order to fully understand what happens following the departure of the projectile from the weapon, a brief introduction to interior ballistics is in order. Figure 1 shows the general characteristics of an artillery projectile.

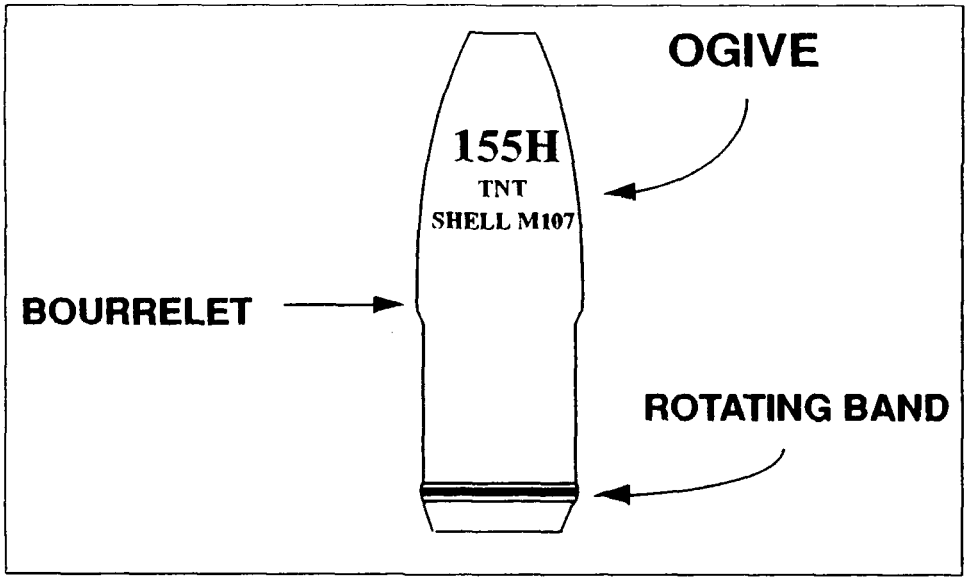

Figure 1: Separate-Loading Artillery Projectile (Shown Without Fuze).

Interior ballistics is the study of the processes that take place inside the bore of a weapon. In general, these processes are responsible for placing the projectile in motion, and to a large extent determine the overall behavior of the projectile during flight. The majority

of ground-based weapons in the United States armed forces are spin-stabilized. That is, their projectiles revolve rapidly about their longitudinal axis of symmetry while in flight. The spinning of the projectile provides stability to the trajectory, as a result of the gyroscopic effect. The spin is developed in the following way: when the projectile is rammed into the breech, the rotating band (in the case of most artillery shells) or the widest part of the round (in the case of munitions lacking a rotating band) contacts the lands and grooves of the bore. When the weapon is fired, the force developed by the rapidly expanding gases of the burning propellant push the projectile forward, causing the rotating band to be engraved by the rifling of the tube. As the projectile travels the length of the tube, spin is imparted as it twists over the lands and grooves. Thus when the round exits the tube, its motion has both translation and rotation characteristics.

B. NEWTONIAN MECHANICS

The discipline of classical mechanics provides us with the simplest method of modeling the motion of a projectile through space. The term "projectile motion" is usually applied to the motion of an object acting solely under the influence of the force of gravity. The position of an object that moves in this fashion may be described as a function of time in a Cartesian coordinate system by the following well-known formulae:

$$x(t) = x_0 + v_{x_0} t + \frac{1}{2} a_x t^2 \qquad \text{(Eq 3.1)}$$

$$y(t) = y_0 + v_{y_0} t + \frac{1}{2} a_y t^2 \qquad \text{(Eq 3.2)}$$

$$z(t) = z_0 + v_{z_0} t + \frac{1}{2} a_z t^2 \qquad \text{(Eq 3.3)}$$

Since we assume that the only force acting upon the projectile is the force due to gravity, we can then conclude that $a_x = a_z = 0$, and that $a_y = g$, where g is the standard gravitational acceleration of the Earth. Equations 3.1, 3.2, and 3.3 therefore reduce to:

$$x(t) = x_0 + v_{x_0} t \quad \text{(Eq 3.4)}$$

$$y(t) = y_0 + v_{y_0} t + \frac{1}{2} g t^2 \quad \text{(Eq 3.5)}$$

$$z(t) = z_0 + v_{z_0} t \quad \text{(Eq 3.6)}$$

The fact that these equations have only one independent variable, namely t, implies that the position function can be implemented in a computer program with a minimum of parameter passing. This means that such a function can be called many times throughout the course of a computer graphics program's screen refresh cycle without greatly increasing the program overhead. For these reasons, it is clear that Equations 3.4 - 3.6 are very well-suited for use in a real-time simulation requiring minimal delays in calculating object position.

While these equations of projectile motion have the desirable characteristics of being straightforward and rapidly computable, they are highly simplified representations of a complex phenomenon. As such, their use is limited in applications which require a high degree of correspondence between physical reality and the simulated environment, such as weapons prototyping or tactics evaluation. Table 2 shows a comparison of values obtained using this technique and values from U.S. Army artillery firing tables (Department of the Army, 1983).

TABLE 1: SIMPLIFIED MODEL DATA VS. FIRING TABLE DATA

Charge[1]	Quadrant elevation (mils)	Observed impact (deflection, range)	Firing table (deflection, range)	CEP (meters)[2]	Error (number of CEPs)
3W	266.67	(4, 5513)	(22, 4056)	10.66	136.7
3W	533.33	(6, 8794)	(84.5, 6588)	17.49	126.2
3W	800.0	(7, 10000)	(207, 7300)	21.35	126.8

TABLE 1: SIMPLIFIED MODEL DATA VS. FIRING TABLE DATA (CONTINUED)

Charge[1]	Quadrant elevation (mils)	Observed impact (deflection, range)	Firing table (deflection, range)	CEP (meters)[2]	Error (number of CEPs)
7W	266.67	(5, 17490)	(69, 8983)	9.8	868.3
7W	533.33	(9, 29522)	(220, 13154)	14.97	1093.5
7W	800.0	(10, 33127)	(441, 14700)	16.73	1101.7

1. Propellant charge is identified by a *zone* (a numeral from one to eight, indicating the general magnitude of the charge), and a *model* (a particular size and shape of propellant grain). In this case, the model is M4A2 white bag.
2. Circular Error Probable.

C. MODIFIED POINT-MASS MODEL

The assumption that no forces other than gravity act upon a projectile in free flight was empirically shown to be an overly broad simplification. The fact is that several other forces do act on spin-stabilized projectiles throughout their trajectory, namely:

1. Air drag.

Projectiles are not perfectly smooth. Defects in the casting of the projectile casing, gouges in the area of the rotating band caused by contact with the forcing cone of the bore, and variations in the formation of the molecular matrix of the projectile skin are all perturbations which impede the flow of air. As a result, the motion of the projectile through the air generates a shearing force as the surface of the round slides past the body of air that surrounds it.

2. Lift.

The angular deflection in the vertical plane between the longitudinal axis of the projectile as it leaves the barrel and an imaginary line that passes through the base of the trajectory is called the *quadrant elevation*. At point of launch, the forward portion of the round's surface area is presented such that as it moves through the fluid medium, the air

moves at differing velocities over the upper and lower portions of the projectile. This difference in air velocity creates a pressure differential that imparts a corresponding force to the projectile, known as *dynamic lift*. Increasing quadrant elevation results in an increase in lift, in much the same way as increasing the angle of attack of an airfoil does.

3. Equilibrium yaw.

As the projectile negotiates the rifling of the bore in its travel during launch, the *bourrelet* of the round rides over the lands and grooves. Nicks and pits in the lands, as well as slight imperfections in the machining of the bourrelet exert impulsive forces which cause the round to precess as it spins. This precession eventually damps out, but results in a permanent deflection from the projectile's original orientation. This difference produces a lateral force in a fashion similar to the way that lift exerts a vertical force on the round. The change in the trajectory resulting from this lateral force is called *drift*.

4. Coriolis effect.

Many conditions used in the development of equations of motion for freely falling bodies assume fairly short times of flight. A classic example is that of a baseball. However, in the case of military ammunition, ranges are typically very large, sometimes on the order of tens of kilometers. Depending on the quadrant elevation, times of flight can be as long as a minute or more. As a result, the rotation of the earth has an effect on the perceived trajectory, in that the time differential between launch and landing is large enough to make the radial motion of the earth noticeable. The effect of this rotation is to cause an increase or decrease in the observed achieved range, depending on the latitude of the weapon.

The point mass trajectory model was developed by Lieske and Reiter to provide a means of calculating artillery trajectories having a computational burden less than the complete rigid body model, yet still incorporating the effects of the forces described above. (Lieske & Reiter, 1966, p. 19). The modified point-mass model was later developed to incorporate the effects of thrust from rocket-assisted projectiles.

According to the modified point mass model (Lieske, 1973, pp. 7-8), the equation of motion for a free-flight projectile is given by:

$$\frac{d\vec{\mathbf{u}}_i}{dt} = -\left\{\frac{\rho\pi m_r v_{i-1}\left[C_D + C_{D_{\alpha^2}}(Q\alpha_e)^2\right]}{8Cm}\right\}\vec{\mathbf{v}}_{i-1} +$$

$$\left(\frac{\rho\pi d^2 L v^2_{i-1}}{8m}\right)(C_{L_\alpha} + C_{L_{\alpha^3}}\alpha_e^2)\,\vec{\alpha}_{\mathbf{e}} + \vec{\mathbf{g}} + \vec{\Lambda} +$$

$$\left(\frac{-\rho\pi d^3 pQC_{N_{P_\alpha}}}{8m}\right)(\vec{\alpha}_{\mathbf{e}} \times \vec{\mathbf{v}}_{i-1}) \qquad \text{(Eq 3.7)}$$

where $\vec{\mathbf{u}}$ is the velocity of the projectile with respect to the ground, $\vec{\mathbf{v}}$ is the velocity of the projectile with respect to the air, and $\vec{\alpha}_{\mathbf{e}}$ is the estimate of the yaw of repose at the *(i-1)*th time step. See Appendix A for a complete listing of the meanings of the remaining variables.

Since this algorithm is intended for use in a real-time, graphics-intensive application, the following assumptions are made to simplify Equation 3.7:

- Mass of the projectile is constant ($m_r = m$).
- Coriolis effect is negligible ($\vec{\Lambda} = (0, 0, 0)$).
- The force of gravity acts only along an axis that is perpendicular to the trajectory.
- No winds are present ($\vec{\mathbf{u}} = \vec{\mathbf{v}}$).

These assumptions yield the following equation of motion:

$$\frac{d\vec{\mathbf{v}}_i}{dt} = -\left\{\frac{\rho\pi v_{i-1}\left[C_{D_0} + C_{D_{\alpha^2}}(Q\alpha_e)^2\right]}{8C}\right\}\vec{\mathbf{v}}_{i-1} +$$

$$\left(\frac{\rho\pi d^2 L v^2_{i-1}}{8m}\right)(C_{L_\alpha} + C_{L_{\alpha^3}}\alpha_e^2)\,\vec{\alpha}_{\mathbf{e}} + \vec{\mathbf{g}} +$$

$$\left(\frac{-\rho\pi d^3 pQC_{N_{P_\alpha}}}{8m}\right)(\vec{\alpha}_{\mathbf{e}} \times \vec{\mathbf{v}}_{i-1}) \qquad \text{(Eq 3.8)}$$

Using these same assumptions, $\vec{\alpha}_{\mathbf{e}}$ is given by:

$$\vec{\alpha}_{\mathbf{e}_i} = \frac{-8I_x p}{\rho\pi d^3 C_{M_\alpha} v_i^4}(\vec{\mathbf{v}}_i \times \frac{d\vec{\mathbf{v}}_i}{dt}) \qquad \text{(Eq 3.9)}$$

p, which is the magnitude of the velocity with which the projectile rotates about its longitudinal axis is obtained by integrating:

$$\frac{dp}{dt} = \frac{\rho\pi d^4 pC_{l_p} v}{8I_x} \qquad \text{(Eq 3.10)}$$

The initial spin velocity, p_0 is given by:

$$p_0 = \frac{2\pi v_0}{bd} \qquad \text{(Eq 3.11)}$$

Equation 3.10 is a linear first-order differential equation in t, which may be solved readily. For the sake of notational simplicity, let

$$a = \left(\frac{-\rho\pi d^4 C_{l_p} v}{8I_x}\right) \qquad \text{(Eq 3.12)}$$

The general solution then is given by: (Eq 3.13)

$$p(t) = ce^{-\int(-a)\,t\,dt} \qquad \text{(Eq 3.14)}$$

$$p(t) = ce^{at}$$

Substituting the initial condition $p(0) = v_0$ gives the particular solution, namely:

$$p(t) = \frac{2\pi v_0 e^{at}}{bd} \qquad \text{(Eq 3.15)}$$

D. IMPLEMENTATION

An object-oriented approach is used in the implementation of the modified point mass model. This allows for an intuitive understanding of the code used in the simulation. A summary of the major object classes follows.

1. Class WorldObject.

This is the base class for objects that are intended to have some physical representation in the virtual world. As a result of the pure virtual member function *draw()*, it is an abstract class, and thus can only be used to derive additional subclasses. Very rudimentary levels of functionality are provided, including the ability to set and report object location and orientation, as well as a tagging scheme to uniquely identify objects without need of using an enumerated type.

2. Class Particle.

Originally designed for use in modeling particle systems, this class turned out to be equally suitable in representing projectile objects. It is derived from class **WorldObject**, and is polymorphic by the addition of data elements pertaining to the object's physical characteristics, such as initial velocity and lifetime, and by its overriding definition of *draw()*. The definition of this function determines the visual characteristics of the particle. By default, objects of type **Particle** are displayed using the Naval Postgraduate School Object File Format (NPSOFF) utilities. See (Zyda, Wilson, Pratt, and Monahan, 1991) for a detailed development of the capabilities of NPSOFF, and (Wilson, September 1992) for a powerful superset of object-oriented extensions to the original design. The remaining attributes of this class are discussed in detail in Chapter IV.

3. Class Ballistic_coefficient.

The equations employed by the modified point mass model make use of a number of dimensionless power products that each capture the state of some physical characteristic of the projectile's flight at a given time. All were determined experimentally, and most are functions of mach number. Some are also dependent upon the quadrant elevation. The intervals of mach number over which each aerodynamic attribute is defined vary for each value, and an updated value is needed for each iteration of the main equation of motion. It is therefore convenient to group sets of these parameters that are applicable to a given mach number into a single data structure. Objects of class **Ballistic_coefficient** contain storage for each of the following attributes:

- Yaw drag.
- Drag force.
- Lift force.
- Overturning moment.
- Magnus force.
- Spin damping moment.
- Yaw lift force.
- Ballistic coefficient for reference mass.
- Lift factor.

The last two items are not coefficients which characterize a ballistic trajectory; rather, they capture various physical aspects of the projectile itself that relate to its form and mass distribution.

4. Class Physical_coefficient.

Because the intervals over which the coefficients that combine to form a particular aerodynamic product are variable in size, an adaptive technique is necessary to extract the appropriate value, given the entry parameter of mach number. Class **Physical_coefficient** provides a means of encapsulating one interval of coefficients and the ceiling mach number for which that set is valid. A detailed explanation of the method used to calculate the products will be given in Section 6.

5. Data structures.

a. Static parameters

The ballistic coefficient and the lift factor used in the equations of motion are functions of the propellant charge and quadrant elevation, and are independent of its velocity. Since these parameters do not vary over the flight of the projectile, they can be calculated only once at the beginning of the integration, and stored. The entry argument used to determine the coefficients needed to calculate these values is propellant charge. For separate-loading artillery ammunition, charge is characterized by a number from one to eight, and a color, either green or white. Because each combination of these attributes produces a discrete value, charge is a more convenient value to use as a table index than muzzle velocity, which is a floating-point value. Three lookup tables indexed by charge are implemented as arrays of floats used to compute the aerodynamic values.

b. Dynamic parameters

The remaining quantities are functions of mach number, and the intervals over which their determining coefficients are valid must be obtained differently for each one. Therefore, a dynamic structure is used to contain the sets of coefficients. An ordered list of objects of type **Physical_coefficient** is created for each parameter, using a library of container classes that are due to Wilson (Wilson, 1992). The list is ordered by the maximum mach number for which that particular set of coefficients is valid.

6. Algorithm.

a. Initialization

The first step in calculating the trajectories is initializing the dynamic data structures. Instances of objects of type **Physical_coefficient** are defined that correspond to a single set of coefficients needed to calculate a particular aerodynamic parameter over a certain mach interval. The values for these coefficients are taken from (Ballistic Research

Laboratory, 1983, pp. 2-8). These objects are inserted into a list in order of the highest mach number over which the set is valid.

b. Numerical integration

The remaining step of the algorithm is to perform numerical integration of Equation 3.8 over the interval of interest. In general, this will be from $t = 0$ until impact. This is accomplished using a predictor-corrector technique described in (Lieske 1973, p. 16). A predicted value for $f(i)$ is obtained by linear extrapolation from $f(i-1)$ by applying the first derivative over the width of the integration time step. This estimate is used to calculate $f'(i)$ in a similar fashion. These two values are then used to calculate $f''(i)$, which is Equation 3.8. The result is used to calculate corrected values of the first two derivatives by the addition of error-localizing terms that are guaranteed to diminish within each iteration of the computation. The values thus corrected become the next data values used to drive the prediction phase, and the algorithm continues until the end of the interval is reached. Since the ith result makes use of values from the $i-1$th time step, it is necessary to retain the most recently completed values from each iteration.The instantaneous values for p are obtained directly by using Equation 3.15.

The values for the aerodynamic inputs in the above calculation are determined by traversing the lists that were created as described in Initialization When the set is found that is valid over the necessary mach interval, the value for the parameter is determined by the following equation:

$$Param = a_0 + a_1 M + a_2 M^2 + \ldots + a_n M^n \qquad \text{(Eq 3.16)}$$

where a_n is the nth coefficient of the set, M is the mach number, and n is the number of coefficients that make up the set.

A comparison of the results obtained using this variant of the modified point-mass model is shown in Table 2.

TABLE 2: MODIFIED POINT-MASS VARIANT DATA VS. FIRING TABLE DATA

Charge	Quadrant elevation (mils)	Observed impact (deflection, range)	Firing table (deflection, range)	CEP (meters)	Error (number of CEPs)
3W	266.67	(26, 4077)	(22, 4056)	10.66	2.0
3W	533.33	(92, 6555)	(84.5, 6588)	17.49	1.93
3W	800.0	(184, 7299)	(207, 7300)	21.35	1.08
7W	266.67	(63, 8940)	(69, 8983)	9.8	4.43
7W	533.33	(210, 13084)	(220, 13154)	14.97	4.72
7W	800.0	(453, 14657)	(441, 14700)	16.73	2.67

IV. TERMINAL BALLISTICS

A. DEFINITION

Terminal ballistics is the study of the events that take place at the end of a ballistic projectile's trajectory. In the case of military munitions, the desired terminal effect is usually to either impart an enormous amount of kinetic energy to the target, or to produce an explosion.

Because of the difficulty of directing them with precision over extended distances, kinetic energy weapons are mainly useful against low-signature, heavily armored targets. The total energy that a particular projectile could theoretically transfer to a target is given by the ubiquitous formula, $e = \frac{1}{2}mv^2$. The means through which these weapons function then is by accelerating a very dense object to extremely high velocity.

The majority of ballistic weapons therefore have been developed for use against area targets. Upon arrival in the target area, these munitions are designed to produce lethal effects by means of the rapid combustion of volatile compounds.

B. CHARACTERISTICS OF EXPLOSIVE MUNITIONS

Explosive munitions all possess a common attribute: they produce destruction in the target area by converting their stored potential energy into kinetic energy in the form of blast, heat, and light. They are further categorized by the manner in which this conversion is controlled.

1. Chemical energy weapons.

Some projectiles are designed such that upon impact, an intense jet stream of hot gases is projected against the target. The enormously high temperatures of the gases cause portions of the target to vaporize, allowing the jet to penetrate further, which causes damage to the interior of the target area. In order to function in this manner, the explosive elements of the projectile must be formed into a parabolic shape, which focuses the energy converted

by the explosion into the characteristic jet stream oriented toward the target. Thus, these explosives are known as *shaped charge* munitions.

Because shaped charge munitions depend upon concentrating their energy against a particular place on the target, they are chiefly useful against point targets, such as bunkers and armored vehicles.

2. Shrapnel-producing weapons.

The majority of the destructive effect in these kinds of weapons is attributable to the blast component. The volatile elements are surrounded by some form of solid matter that disintegrates into smaller pieces under the forces of the rapidly expanding gases produced by the explosion. The kinetic energy of the gases' expansion is imparted to the particles of shrapnel, which are propelled from the locus of the explosion at high velocity. The shrapnel then collides with objects in the target area, producing the desired effect.

From the virtual world perspective, the shrapnel-producing weapons pose the more interesting problem: how can the visual characteristics of such weapons be modeled? To answer this question, we must examine the results of experimental testing.

Experiments have been conducted wherein a particular munition is detonated, and the final positions of the resultant particles of shrapnel are recorded. The evidence shows that the characteristics of these particles vary with the shape of the projectile, the nature of the confining material, and the type of explosive (Reche, 1980, pp. 170-171). In general, though, the initial velocity vectors of these particles can be assumed to have a normal distribution with respect to their magnitudes, and a uniform distribution with respect to their orientations. The sizes of the particles produced can be assumed to be normally distributed. Assuming these orderings then, a key element in the development of a way to model a munition's terminal effects in a synthetic environment is appropriate selection of the mean and variance of each distribution.

Another important attribute in the visual modeling of exploding shrapnel has been referred to as the *directionality*. This term is used to describe the global orientation of

the particles. It is defined by a principle orientation and a limiting angular deflection known as the *ejection angle* (Reeves, 1983, p. 367). As in the case of velocities, a normal distribution of shrapnel pieces within the ejection angle is expected. For ground bursts, the surface upon which the munition impacts usually forces the principal orientation to be perpendicular to it. Figure 2 shows a schematic of these fundamental quantities.

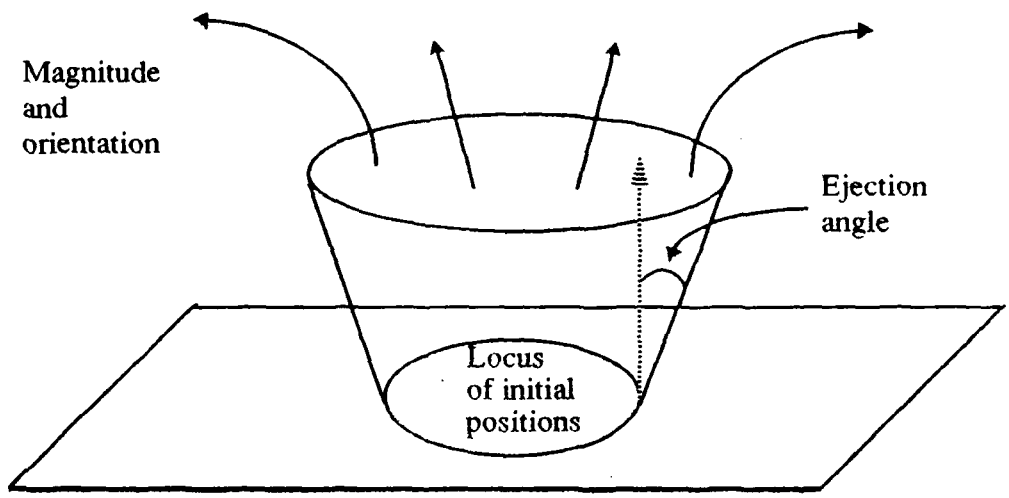

Figure 2: Essential Visual Characteristics of a Shrapnel-Producing Explosion (Ground Burst).

In the case of air bursts the directionality is neutral, since shrapnel is ejected roughly equally in all directions.

C. EXPLOSIONS EDITOR (NPSEE)

As an aid to selecting values for the variables described in the previous paragraphs, the Naval Postgraduate School Explosions Editor (NPSEE) was developed. This is an interactive design tool which provides a means to map out the overall characteristics of an explosion animation sequence. In addition to the global attributes already discussed, NPSEE allows a designer to specify:

- Shape of the region in which particles are initially distributed.
- Density of the initial particle distribution.

- Nature of the particles' individual trajectories (linear or parabolic).
- Initial color of the particles.

As the values for the parameters of interest are adjusted, an on-screen "template" is changed to give the user visual feedback of the effect of a particular action. The parameters are then saved to a file, which can be read in to control the actual animation. The editor is not a keyframing or sequencing device. It does not seek to define exactly what the animated effect will look like; rather, it is a support tool that can be used to test various combinations of the controlling variables.

1. Interface.

For the most part, input to NPSEE is managed through use of a screen-oriented graphical user interface, called NPS Panel Designer (NPSPD) (King & Prevatt, 1990). NPSPD produces an interface that is intuitive and visual. Users provide input by manipulating various controls on-screen with the mouse. The controls used in NPSEE are either buttons or sliders. Pressing the left mouse button while the cursor is over a slider and then dragging the slider to the desired position allows values along a continuous spectrum to be specified. Some controls pertain to parameters that have no easily implemented visual analogue, and thus changing them does not affect the shape that is displayed in the template window. They are nonetheless useful, in that the point-and-click method of input management is usually less tedious and error-prone than typing. Figure 3 shows the user input screen of NPSEE.

2. Parameter encoding.

Visual feedback concerning the effect of changes to the global state variables is provided by means of an iconic shape in the template window. It is intended to present the user with an encapsulation of the effects that the current parameter settings will have on the explosion particle system. This is accomplished by encoding key information in the characteristics of the template shapes that are displayed. For an air burst, the particles are flung out in all directions. Thus, a spherical shape is used to convey this type of dispersion.

In a ground burst, the particles are mainly propelled outward and upward, away from the site of the explosion. A more complex shape is needed to capture the information in this case. For circular ground bursts, a roughly cylindrical shape similar to that shown in Figure 3 is constructed using non-uniform rational b-splines (NURBS). The resulting form conveys information in the following manner:

- The height of the shape determines the mean magnitude of the particles' velocity distribution. It is adjusted using the 'Y-mag' (Y-magnitude) button.
- The angle that the sides of the shape make with the x-z plane determines the degree to which particles are cast outward from the center of the explosion. It is adjusted using the 'Ejection angle' slider.
- The overall scale of the shape determines the size of an individual particle for both ground and air burst. It is adjusted using the 'XZ-mag' slider.

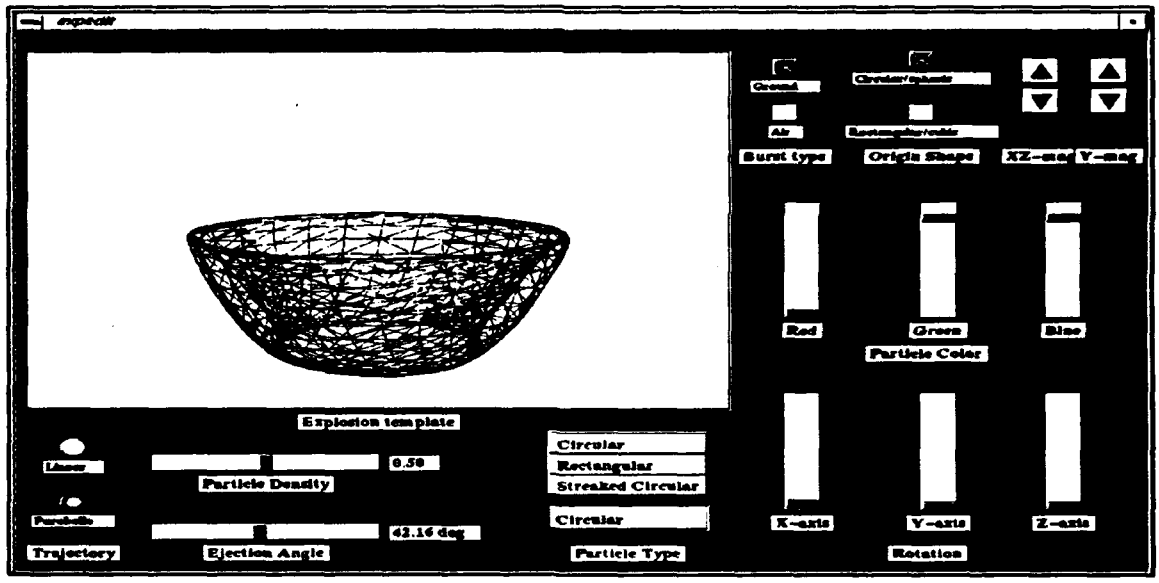

Figure 3: The NPSEE Display.

For the ground burst template, three viewing options are available by pressing the right mouse button to view the main menu. Either the splines that determine the shape, the polygons used to tessellate the surface, or the interpolating surface itself may be shown. Once the variables have been adjusted as desired, they may be saved to a file for use in the animation sequence.

D. PARTICLE SYSTEMS

1. Fuzzy objects.

There exist physical phenomena occurring in nature that are composed of a large number of bodies which are small in comparison to the space occupied by the sum of the bodies taken as a whole. Some examples are clouds, hair, smoke, fog, steam, fire, dust, and snow. The visible manifestation of these objects have a differing cross-sectional density, usually becoming more dense toward the interior of the volume that the object occupies. The diminishing of body density toward the outer limits of the object has the effect of causing the edges to be ill-defined.

Rendering this class of objects in a virtual world poses special problems. The computational burden of determining the motion of each body during the screen refresh period is too great, given the state of the art of computer hardware. This difficulty is exacerbated by objects whose component bodies' visual characteristics also change over time. In general, this category of objects incorporate too much detail to represent convincingly using deterministic methods.

Aside from the common thread of enormous visual detail, most of the entities that comprise these "fuzzy" objects have a dynamic *lifetime*, the period of time throughout which the particles are visible. As an example, consider a cloud of smoke. Over time, the cloud moves about, and is dissipated by the motion of the air, as well as other influences. When a portion of the cloud has thinned past a certain density, that portion appears to an observer to have faded away. The observer will notice that in general, clouds of smoke tend not to fade away at the same rate. The same thing is true of fireworks. Thus there is a need to attenuate a particle's participation in the simulation over time.

2. Management of complexity.

A *particle system* is an abstract computational structure that encapsulates some of the complexity of this class of objects. They are collections of either other systems, or atomic particles. They necessarily incorporate some form of stochastic algorithm for

managing the aspects of the object's appearance and behavior that cannot be accomplished explicitly. The attributes that are frequently governed in this way are position, velocity, size, color, shape, and lifetime. In cases where the object is an aggregate of particle systems, it is often convenient to establish a hierarchical relationship between them. In this way, some of the subordinate elements' global qualities can be encoded into their positions in the hierarchy. This simplifies the task of managing attributes that change over time.

Exactly *how* to determine the state of such characteristics as a function of time is closely tied to the physical phenomenon being modeled. It may be possible to discover an analytic function that will suffice. An example of this technique would be use of equations 3.1 - 3.3 to compute the position of a projectile. Alternatively, a linear interpolation might be useful, as in

$$f(t_n) = f(t_{n-1}) + (\frac{df}{dt} \times \Delta t) \qquad \text{(Eq 4.17)}$$

Most problem domains require some degree of empirical evaluation to determine the most appropriate technique. In situations where a probabilistic distribution of values is required, a variant of the technique given by Reeves (Reeves, 1983, p. 361) may be suitable:

$$f(t) = \mu_f + (Rand() \times a\sigma_f) \qquad \text{(Eq 4.18)}$$

μ_f and σ_f are the mean and variance of the desired distribution, respectively. *Rand()* is a function that returns some floating-point value between 0 and 1. Clearly, it is important that the implementation of *Rand()* be chosen to reflect any modalities that might be present in the way that the attribute varies in reality. The method of constraining *f(t)* to lie within $\pm a\sigma$ of μ is useful when *f(t)* models a physical process that has predictable boundary conditions.

E. IMPLEMENTATION

The notion of a class in C++ is an abstraction that is well-suited to the implementation of an explosion particle system model. The ability to encapsulate data elements, as well as defining initialization parameters within object constructors turns out to have direct analogues to the particle system metaphor. By conceptually equating a particle with an object, the logic of the main routine that accomplishes the motion of the particles is greatly simplified.

The following attributes are handled in a stochastic fashion:

- Initial positions.
- Muzzle velocities.
- Orientation and quadrant elevation used to calculate the particles' trajectories.

The locations of the particles are determined such that they lie initially within the boundaries of a circular region. The formula given by Equation 4.18 is used to generate a placement set that is square; these values are subsequently clipped to lie within a circle of an appropriate radius. The distribution of these locations is roughly uniform with respect to the circle's area.

1. Visual characteristics.

Each individual particle in the simulation is a tetrahedron. The NPSOFF utilities discussed above were used to render these objects. The original choice was a two-dimensional circle, chosen for its simplicity and consequent speed of rendering. The resulting animation, while very fast in terms of frames per second, was not very persuasive. To make the picture interesting, it seemed that a form that had shape in three dimensions was required. The next logical choice was a polygonalized sphere. The sphere was fairly expensive to render, and it also had the disadvantage of being unbelievable as a form meant to represent a piece of shrapnel. The tetrahedron is a reasonable compromise between these concerns.

2. Data structures.

In this simulation, a system of particles is implemented as a class containing a doubly-linked list of atomic objects. Once again, good use is made of Wilson's NPSCL to obtain this functionality, and to maintain the object-oriented motif of the program. Three objects of this class are instantiated: one holds the state of the system as it is being rendered by the graphics hardware; one holds the state of the system as updates are calculated for the succeeding frame; the last is used as a buffer between the rendering and calculation states.

3. Algorithm.

Upon start-up, each particle in the system is assigned characteristics of position, velocity, lifetime, and quadrant elevation. Using a simplified dynamics model, the projected location of each particle is calculated instantaneously based on its current location and velocity vector, and rendered accordingly. The temporal resolution of this calculation was chosen so as to minimize jerkiness, but still give a faithful impression of a ballistic trajectory. On an Iris 4D/240 VGX, particle counts of up to approximately one hundred result in satisfactorily realistic motion. Every particle acts under a force designed to simulate the earth's gravity, as do particles of shrapnel in a real explosion. Collision with the ground is detected by examining the y-component of the particle's position. An elastic point-to-point collision is simulated, with a linear damping of the original particle energy.

Parallelism is exploited by partitioning the program into two MIMD processes. One process (the "producer") is tasked to calculate succeeding positions for each particle. The remaining process (the consumer) reads the output from the producer, and displays particles at the corresponding positions. A two-buffer scheme due to Dr. Michael Zyda is modified to incorporate the dependency upon previous particle velocities, and used to enforce mutual exclusion between the shared data. Some preliminary testing showed that the program's performance could probably be improved by managing a queue of buffers, albeit with attendant added complexity. Figures 4 and 5 show the resulting output.

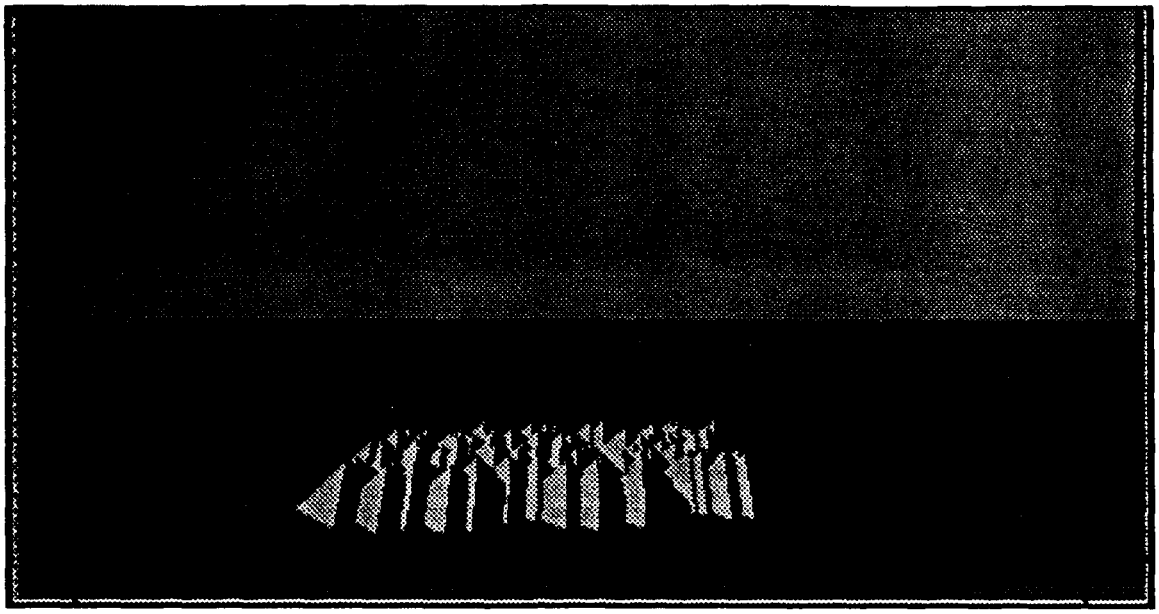

Figure 4: Initial Particle System Disposition ($t = 0.0$, $n = 100$).

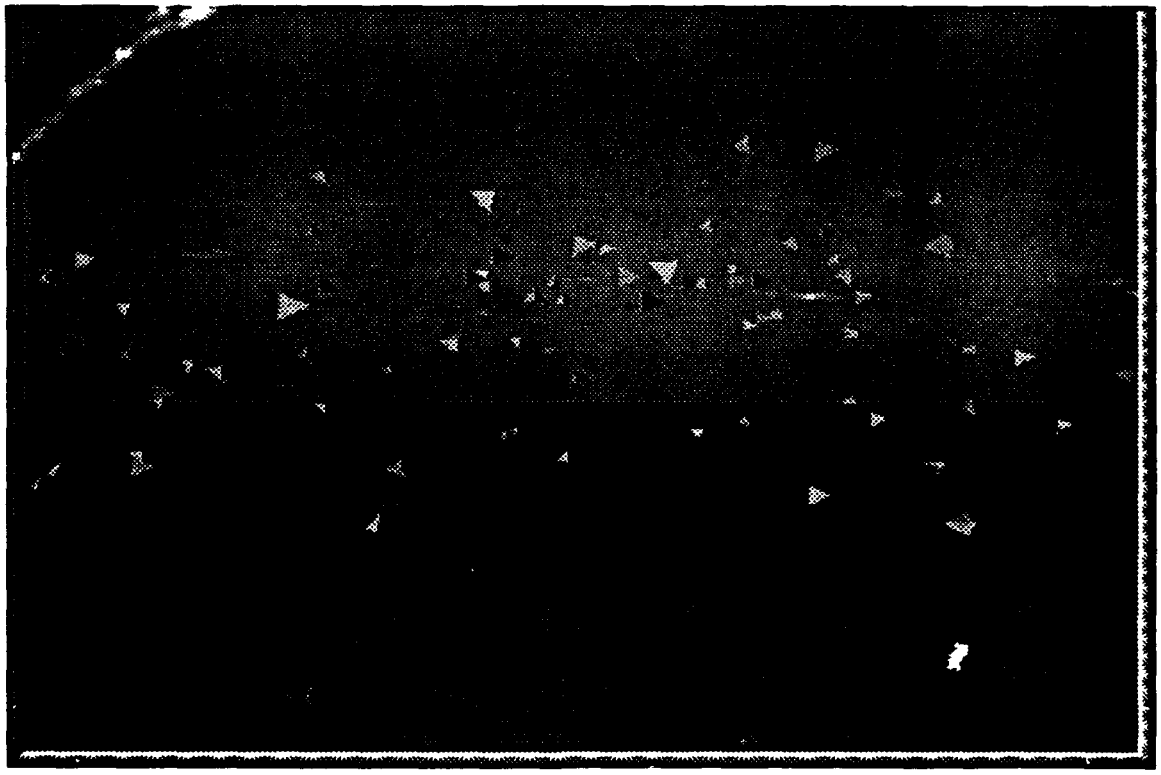

Figure 5: Development at $t = 2.14$.

V. THE AUTONOMOUS FORWARD OBSERVER

A. AUTONOMOUS AGENTS

1. General.

The creation of a computer program that successfully imitates the human capacity for perception and thought has been the holy grail that computer scientists in the field of artificial intelligence have pursued since the beginning of the discipline. According to some at least, little real progress has been made toward that end (Wilkes, 1992, p. 17). Certainly inroads have been made in the intensely studied area of computer vision. Edge detection, region-growing, and stereoscopy are now well-developed methods for allowing a hardware and software system to "see". Also, work in the fields of digital signal processing and natural language understanding has enabled computer systems to "hear" and respond to the human voice (Luger and Stubblefield, 1989, pp. 378-379). It would appear that the harder problem has been defining a means by which the data thus received can be represented and reasoned about. Many such methods have been proposed; see (Jackson, 1990) for a comprehensive treatment of the subject of knowledge representation. The fact that no single method has been shown to be useful in the general case of reasoning over many wide problem domains suggests that perhaps the definitive work in this area has yet to be undertaken.

In the conduct of interactive computer simulations involving the combat of ground forces, it is frequently desirable to provide a capability to enable entities that can automatically emulate the actions of a particular group of participants from either side. This reduces the number of humans required to conduct the simulation, which is a generally desirable characteristic. This capability is especially useful when special skills are required to interact in the roles concerned. To this end, an expert system could be constructed with the reasoning abilities necessary to simulate the cognitive contributions of its human counterpart.

Portraying the presence of autonomous forces in a virtual world is related to the problem of creating autonomous robots that can interact with human beings, but with an important difference: the sensory outputs that are available to humans as well as the inputs themselves must be simulated. How to simulate the various communications channels, as well as what information ought to be provided to an autonomous entity has been the subject of some debate.

2. Global vs. local orientation.

The research in the area of intelligence simulation in autonomous agents has focused mainly on the development of efficient algorithms that manipulate an internal symbolic representation of the world knowledge available in order to pursue a pre-determined set of goals (Maes, 1990, p. 49). Known as "traditional" AI, this approach has been considered by some to be problematic for certain dynamic, real-time environments, especially in the face of contradictory inputs (Pylyshyn, 1987). An alternative paradigm has been developed in recent years which emphasizes the tight coupling of perception to action, without the need for deferring a response until the environment's global state can be determined. An example of this localized orientation is the subsumption architecture (Brooks, 1989, p. 692). Research with autonomous robots has shown that surprisingly complex behaviors can appear in systems that are programmed to react to sensory inputs with little recourse to global strategies. There is no reason to expect that the same would not be true of autonomous agents in the context of a virtual world.

3. Suitability for synthetic environments.

Autonomous agents designed using the non-traditional mode have several characteristics that are desirable in terms of their implementation in the context of a virtual world. First, the presence of intelligent agents in a simulation implies an increase in program overhead having computational complexity of at least $O(n)$, due to the requirement to emulate human sensory systems. Each agent extant in such a system must be updated periodically regarding what can be seen, what can be heard, whether or not it

has collided with another object, etc. It is therefore necessary to minimize communications to and from agents, if graphical representations of their actions are to be rendered in real time. Localized architectures allow the decision-making modules to be incorporated at a low level, obviating the need for passing information received from the sensory modules to a global planning process. Simulations involving the participation of large numbers of agents (human or otherwise) can rapidly overwhelm the computing capacity of even the most powerful systems. A movement toward distributed knowledge and function seems inevitable as the expectations of complex behaviors in autonomous agents rises with the acceptance and employment of computer simulations in general.

The object-oriented paradigm of programming is widespread in the implementation of virtual worlds. Due to its built-in information-hiding features, the object-oriented technique is a natural choice to implement the decentralized decision-making model. Indeed, it is assumed that complete world knowledge is not available to each entity in such an environment. Not only does this have the effect of minimizing information transmission, it encourages the development of a standardized interface that allows interaction between agents that occupy a simulation.

As graphical simulations become more sophisticated, increasing user expectations of realism will require the interaction of hundreds, perhaps thousands of simultaneous participants. It seems unlikely that advances in computer architecture will result in the creation of a single device that would be able to accommodate the computational load that such a simulation would entail, at least not in real time. Therefore it is clear that any simulation that seeks to maximize the appearance of realism will necessarily involve the cooperation of multiple networked computers. This suggests that the overall nature of such a simulation will be highly parallel and decentralized. The non-traditional approach to the design of intelligent agents is well suited to such an implementation.

4. Belief systems.

Another design decision that implementors of autonomous entities face is how to determine just what it is that a particular agent knows, and what it believes. Note that in this context *belief* is the agent's perception of fact regardless of its truth in the absolute sense; *knowledge* is awareness of absolute truths coupled with a belief to that effect. To some extent the manner in which knowledge and beliefs are gained is related to the choice of orientation discussed above. The orientation of the overall decision-making paradigm (global planning vs. localized reaction to stimulus) will affect *what* an agent is permitted to know about his environment.

When a goal of an agent's participation is to imitate the actions of some human counterpart (as is usually the case), it is most often considered undesirable that the agent have knowledge about the world that is complete and accurate. This is true, of course, because human beings are not all-knowing. Not only is it the case that we are not oracles, but even when provided with explicit, accurate knowledge, humans cannot always be counted upon to make logical decisions. This is attributable to a variety of internal and external factors. Most of the external factors bearing upon judgement have to do with the attenuation of perception. For example, a soldier that is wearing night vision goggles (NVGs) perceives a reconstruction of the actual world image that is illuminated in such a way as to cause the appearance that every object is some shade of green. Thus his sense of vision is sharply attenuated. If the same soldier, still wearing NVGs, was driving a truck in an urban area where there were color-coded traffic lights, we would perhaps expect that his imperfect knowledge about the color of the lights that he sees might lead him to take an inappropriate course of action, such as proceeding through an intersection when the traffic light is red. (For the sake of illustration, we neglect the fact that an additional relational encoding exists with respect to the positions of the lights in the vertical plane.)

Thus we conclude that in order to preserve the appearance of *human* intelligence, the sensory apparatus of autonomous agents must somehow be distorted. However, not just any means of distortion will suffice. It is a safe assumption to say that since the autonomous

agents we are discussing are meant to simulate the actions of human analogues, then in general they are programmed to respond to sensory stimuli in the same ways that human beings do. Therefore it is not reasonable, for example, to expect a program that controls an agent's responses to consider the possibility of a cow that it encounters producing an automatic weapon and taking him under fire (although stranger things have happened in the wide world of virtual realities - see (Zyda and Pratt, 1992)). In iecting a filter, it must be understood that too large a deviation from the actual inputs might appear obviously ludicrous, and destroy the illusion of realism. Sensory deformations that give the impression that a law of physics has been violated, or that vastly contradict the agent's previous experience are the most likely to produce inappropriate behavior in autonomous agents, because it is precisely by these anticipated events that its responses are defined.

Bhargava and Branley have explored this issue, and discovered a series of filters that work well with autonomous agents in a virtual world (Bhargava and Branley, 1992, p. 14). Their belief function is designed so that it varies logarithmically, and may be clamped to maximum and minimum bounding values that have correspondence to the appropriate limits in the real world. They also describe a way that the function may be applied so as to accommodate contradictory inputs, using Dempster-Shafer theory (Shafer, 1976). Their method was used to distort information that was given to a platoon of vehicles acting autonomously within NPSNET, with very satisfactory results (Branley, 1992).

B. AUTONOMY IN NPSNET

All vehicles in NPSNET behave autonomously, with the exception of a single "driven" vehicle per workstation. The driven vehicle can be changed at any time during the simulation. The other vehicles are programmed to meander about the terrain at varying speeds and directions, reversing their direction when either the virtual ocean or the end of the terrain database is encountered. They have a sensory capability that makes them aware of other vehicles that are firing at them, and can interact with other vehicles by either returning fire upon them, or running away from them.

Culpepper implemented a module in NPSNET that allowed for a platoon of autonomous armored vehicles to operate in a goal-oriented fashion (Culpepper, 1992). They obtained mission and subgoal assignments dynamically in response to changes in the environment, such as the presence of land features, and incoming weapons fire. These vehicles also have the capability of returning fire, as well as initiating offensive firing in a direct mode.

The autonomous entities discussed so far are limited to firing upon only those vehicles with which they have direct visual contact. This is by conscious design, not oversight; the same thing is true of vehicles operated by human beings. However, there is a way that fires can be brought onto objects by entities beyond the targets' line of sight.

C. DIRECT VS. INDIRECT FIRES

As in real combat, participants in the conduct of direct fires are at significant risk of being acquired as targets themselves by the vehicle that is being fired upon. An alternative to enduring this risk is the employment of *indirect* fires, as the field artillery does. This technique may be used to lob munitions onto an enemy along an arcing path, rather than propelling them straight toward the target. It has the advantage of being unhindered by intervening objects such as terrain features, as well as reducing the previously mentioned risk of revealing the firing element's location to the enemy.

Along with the advantage of the shooter's security comes an equally significant disadvantage: because of the characteristically high muzzle velocity of military weapons, the range to target in general is beyond the capability of the firing entity's ability to observe. Thus, it is usually impossible for an individual using a weapon that fires in a purely indirect mode to acquire targets on his own that he can shoot at effectively. As well, he cannot evaluate the effects of the fires he produces, since they land beyond the limits of his vision. It is therefore clear that in order to employ such weapons with any efficiency, an additional player must be involved.

D. THE ARTILLERY FORWARD OBSERVER

The person responsible for acquiring targets, communicating their location to the firing elements, and evaluating the effects of their fires against the targets is the *forward observer* (FO). Since the FO is not actually firing, it is possible for him to remain undetected while attacking the targets. In order to perform his mission, the FO must be able to perform the following cognitive tasks:

- Be aware of his own location.
- Sense the presence of other entities within range of his sensory capabilities, and determine their location.
- Determine the status of other entities as either friendly or enemy.
- Communicate the location of entities determined to be enemy to the firing elements.
- Sense the presence of munitions arriving in response his requests.
- Evaluate the effects of the fires noted, and take appropriate action based on that evaluation.

A more sophisticated model might include tasks such as taking actions to preserve the observer's own life, should the need arise. Since our main interest is in reproducing the capabilities that are necessary to permit the howitzers to fire in indirect mode, we will neglect such issues.

E. IMPLEMENTATION

All of the tasks listed above imply the receipt or transmission of sensory information as a prerequisite to the reasoning process. Since the necessary information is represented in NPSNET as C data structures, the modules that simulate the sensory processes were implemented in the C++ programming language. Once the sensory information is processed into data, it is communicated to an appropriate reasoning module written in CLIPS. These modules (*rules*, in CLIPS parlance) contain the logic needed to perform the observer's cognitive tasks. As a result of the functioning of these rules, the appearance of an autonomous entity that behaves in the manner of an artillery forward observer is obtained.

The observer is made aware of certain key default data at program start-up by means of a *deffacts* construct, and additionally a rule that creates instances of the howitzers and the fire direction center. The deffacts informs the observer of his location, and the kinds of vehicles he is to consider as hostile. The howitzer objects are made aware of their initial locations as well.

The fire direction center (FDC) is an entity whose participation in real indirect fires is crucial, but curiously turned out to be of minimal significance in this implementation. In tactical artillery units, personnel in the fire direction center are responsible for calculating the orientation of the howitzers necessary to hit the target, given the locations of the howitzers and the target. Since this is a very mechanical, algorithmic process, these calculations are performed in the C++ modules, and the FDC object is maintained in the CLIPS modules mainly for the sake of architectural integrity of the indirect fire paradigm.

After initialization, NPSNET has a main driver process that iteratively determines changes to the status of objects (in terms of their location and appearance), and renders the currently visible scene on the user's display. Code is inserted into this process such that with every iteration:

- The CLIPS modules are advised of the current locations of the howitzer and the observer.
- A determination is made of what vehicles (if any) can actually be seen from the observer's position.
- The CLIPS modules are advised of the locations and types of the vehicles that can be seen, as well as the locations of impacting artillery.
- The CLIPS inference engine is allowed to run to completion.

The determination referred to in the third point above is distinctly non-trivial. It involves a two-tiered examination of every vehicle in the simulation. First, each vehicle is tested to see if it is close enough to be seen with the naked eye. In this implementation, this distance is assumed to be no greater than four kilometers. All vehicles that pass this first test are examined for the possible presence of obstructions between the observer and the vehicle. A ray is traced between the observer and the target by determining the parametric

equations of a line between those two points. The level of the ground is calculated at equidistant intervals along each ray to see if any terrain features intervene. The locations and types of the vehicles that pass both tests are asserted to CLIPS as "contacts".

The CLIPS modules examine each contact to see if it meets the criteria to be considered a target. If it does not, the contact fact is simply retracted. If the contact is a hostile one, the contact fact is changed to a target fact. Upon assertion of the target fact, rules determine the location of the target with respect to the howitzers, and calculate the necessary deflections and elevations needed to cause the rounds to land as close to the target as possible. A message is sent to the howitzer objects to slew the tubes to the correct azimuth and elevation, and to fire themselves, which results in the transmission of data back to NPSNET to do so. When the rounds finally land, the observer sends a request to NPSNET to determine if the vehicle that it was shooting at was in fact killed. If it was killed, then the observer causes the guns to cease firing. If it was not killed, the observer causes the guns to fire again. If the target is in motion between the time it is detected and the howitzer fires, the observer recalculates the mission each time with a new target location. This technique differs from the doctrine currently in use in the U. S. Army; a real observer would make adjustments in terms of deviations from the locations of previous impacts, rather than absolute locations.

F. LIMITATIONS

As with all simulations of human intelligence, there is some discrepancy between the operation of this implementation and the way a real forward observer behaves. In particular:

1. Visibility determination.

The ray-tracing visibility test is not precise. In order to save processing time, the elevation of the ground is examined at no more than one hundred locations between the observer and the target. Since the established maximum range of the observer's vision is

approximately four thousand meters, this means that terrain features of less than forty meters breadth (measured parallel to the observer-target line) may not be detected.

2. Range resolution.

The increments with which the howitzers can adjust their elevation is limited by the length of the lookup table. At the moment, the resolution is a minimum of 250 meters. This makes fast-moving targets difficult to kill. A numerical root-finding algorithm that solves Equation 3.8 in terms of quadrant elevation would reduce the coarseness of this parameter.

3. Observer behaviors.

The observer is limited in his actions upon detection of a target. At this point, the observer's operant philosophy could be summed up as, "If an enemy vehicle is detected, bring fires upon it until it is dead." Real observers are somewhat more flexible in their pursuit of the mission. While the format of the facts pertaining to mission assignments requires a selection from one of the three classical field artillery missions, only the adjust fire mission is supported by the rule base.

The observer is incapable of sensing the need for requesting high-angle fires, as would be required in a case where the target was masked from the firing platoon by high terrain.

4. Battle damage assessment.

The kill mechanism of a bursting projectile is represented as a binary, all-or-nothing affair. There is no provision for assessing levels of damage less than complete destruction. For this reason, no attempt has been made to include logic that selects an optimum shell-fuze combination based on the target characteristics.

5. Belief modes.

The sensory data received by the observer, and consequently by the FDC to compute the firing data, is perfect. No real-world signals attenuation such as fog, radio

static, fatigue, is portrayed. This is unrealistic, because many of the subtasks that the observer has to perform depend directly on the quality of information available to him. Examples of such subtasks are target acquisition and target location. Additionally, there is no provision for possibilities such as an unreliable observer, or mistakes in judgement.

Along similar lines, an assumption here is that no misunderstandings between the FDC and the howitzer crews take place, and that the crew orients the weapon without error. Experience suggests that this is actually an appreciable source of inaccuracy in weapons firing.

These limitations notwithstanding, the implementation has been used successfully in a small combat scenario with satisfactory results. Even considering the primitive nature of the autonomous observer's abilities, a powerful battlefield force can be brought to bear using these techniques during a simulation.

VI. CONCLUSION

Finding a way to realistically represent objects that appear in a particular environment is at the heart of all simulation systems. Military simulators have an especially acute need to find representations that are suitable for use in modeling ballistic projectiles, since these objects are always present in non-trivial combat scenarios. Simulations that seek to model military operations accurately must include a ballistics model that is as faithful to the actual weapon's performance as possible, given the usual speed versus accuracy trade-offs. Lieske's modified point-mass trajectory model was shown to be suitable for implementation within a real-time graphical virtual world for this purpose.

This research also presents a means by which the visual characteristics of the terminal effects of shrapnel-producing weapons can be portrayed. Although the particle images themselves lack detail, they are nonetheless meaningful because the richness of the variations in the motion of the particle system in its entirety can convey an animated sequence that is believable as an explosion. This can enhance the realism of the simulation overall. Additional research is necessary to discover how the effects of the destructive mechanisms in such weapons (i.e., energy transfer) can be similarly modeled.

The field artillery is known in some circles as the "King of Battle", allegedly because it is responsible for the highest number of casualties in the history of warfare. In any event, the ability to bring fires onto a target area that is miles away from friendly lines is considered fundamental to the combat tactics of every modern army in the world. The capacity to simulate the delivery of indirect fires is therefore an important part of combat simulators. The object-oriented approach toward modeling the forward observer taken in this work is desirable because it results in an implementation that functions in much the same manner as its human counterpart. This produces behavior that is convincing, through programming that is easy to understand and modify.

APPENDIX A (VARIABLE DEFINITIONS)

A. NOTATIONAL CONVENTION.

In the equations used in this paper, vector quantities appear in bold type, with a vector symbol directly over the variable, as in $\vec{\mathbf{v}}$. All other variables should be considered to be scalars. When a variable that is shown in non-boldface type without the vector symbol also appears elsewhere in vector format, it should be understood that the *instantaneous magnitude* of the vector is called for. This avoids the somewhat cumbersome notation of the vector's norm, as in $\|\vec{\mathbf{v}}\|$.

B. VARIABLE DEFINITIONS

Equations which incorporate variables that do not make use of the usual SI units for that quantity (e.g., quadrant elevation) are not dependent upon them for their dimensional correctness. The following variables are used throughout this paper:

Variable	Meaning	SI units
b	Barrel rifling twist	Calibers/rev
C	Ballistic coefficient	kg/m^2
C_{D_0}	Drag force coefficient	Dimensionless
$C_{D_{\alpha^2}}$	Yaw drag force coefficient	Dimensionless
C_{L_α}	Lift force coefficient	Dimensionless
$C_{L_{\alpha^3}}$	Yaw lift force coefficient	Dimensionless
C_{L_p}	Spin damping moment coefficient	Dimensionless
C_{M_α}	Overturning moment coefficient	Dimensionless
$C_{N_{p_\alpha}}$	Magnus force coefficient	Dimensionless
d	Diameter of the projectile	Meters

Variable	Meaning	SI units
$\vec{\mathbf{g}}$	Gravitational acceleration of the earth	Meters / sec^2
I_x	Axial moment of inertia	$kg \cdot m^2$
L	Lift factor	Dimensionless
m	Projectile mass	kg
p	Axial spin velocity	Radians/second
$\dot{p}$	Axial spin acceleration	Radians / $second^2$
Q	Yaw factor	Dimensionless
QE	Quadrant elevation	Mils
t	Elapsed time	Seconds
$\vec{\mathbf{u}}$	Velocity of the projectile with respect to the air	Meters / second
$\vec{\mathbf{v}}$	Velocity of the projectile with respect to the ground	Meters/second
$\vec{\alpha}_e$	Yaw of repose	Radians
Λ	Coriolis acceleration of the earth	meters / $second^2$
ρ	Air density	kg / $meter^3$

APPENDIX B (USER'S GUIDES)

A. BALLISTIC TRAJECTORIES

To enable the calculation of ballistic trajectories in NPSNET, specify the "b" option on the command line, for example, "npsnet b". No other actions are required. If you wish to modify the default muzzle velocities for existing weapons, or if new weapons must be added, consult the file ./data/weapons.dat. It is an ASCII text file containing the parameters that define the characteristics of weapons in NPSNET. The function that reads this file ignores comments that appear which use the "C" programming language conventions. The format for entries in the file is given here.

The identification field begins with the character "I". The following entries must be present, in the following order and format:

- char* name (name of the weapon, max 20 characters with no spaces).
- int id_num (number used to identify a particular weapon no two weapons have the same id_num. USSR weapons have numbers starting from 2, US weapons start from 100).

The data field begins with the character "D". The following entries must be present, in the following order and format:

- int range (absolute maximum range, in meters).
- float muzzlevelocity (how fast the round travels as it leaves the firing point, in meters per second).
- int killsize (outside radius from the point of impact of lethal effects, in meters).
- int damagesize (outside radius from the point of impact of non- lethal, but nonetheless noticeable effects in meters.
- char knowledgesize (outside radius from point of impact within which a player vehicle should notice that a round has impacted, in meters).
- char targettype (the type of target that this weapon is typically most effective against. Key:

  ~ "0" = Any target

  ~ "1" = Heavy armor

  ~ "2" = Light armor

~ "3" = Wheeled vehicle

~ "4" = Personnel

~ "5" = Aircraft

~ "6" = Watercraft

- char pathtype (default trajectory employed by the weapon.) Key:

~ "0" = low angle [direct fire]

~ "1" = high angle [indirect fire]

~ "2" = linear [laser beam]

~ "3" = gravity bomb.

Note: At the moment, the entries for killsize, damagesize, knowledgesize, and targettype are not used. Several examples are shown below:

```
I 155mmM109Howitzer 110 /* Non rocket-assisted, Charge 3WB */
D 7300 297.0 100 150 200 1 1

I 203mmM110Howitzer 111 /*8-inch howitzer, non rocket assisted*/
D 16800 370.0 150 200 250 1 1

I 50cal 112
D 1800 100.0 2 5 50 2 0
```

IMPORTANT NOTE: The autonomous forward observer is designed to operate with the firing platoon shooting at a muzzle velocity corresponding to charge three white bag. The reason is that the algorithm for calculating the quadrant elevation for the howitzers to achieve the range necessary to hit the target must make an assumption about the muzzle velocity in order to function, given that the achieved range is a direct function of charge and elevation. If the default muzzle velocity for M109 howitzers is changed to a value different than charge three, the howitzers will not respond properly.

B. NAVAL POSTGRADUATE SCHOOL EXPLOSION EDITOR

The meanings of the graphical controls in NPSEE were described in Chapter IV . In addition, the following keyboard commands are in use:

- Keyboard up arrow, down arrow: viewpoint z-, z+.
- Keyboard left arrow, right arrow: viewpoint x+ x-.

- Keyboard home, end: viewpoint y+, y-
- Keypad up arrow, down arrow (8 and 2): reference point y+, y-.
- Keypad left arrow, right arrow (4 and 6): reference point x-, x+.

C. AUTONOMOUS FORWARD OBSERVER

To enable the autonomous forward observer in NPSNET, specify the "a" *and* the "b" options on the command line, for example, "npsnet a b t". No other actions are required. Upon program start-up, the forward observer's vehicle (an M577 by default), and a platoon of four M109 howitzers will be added to the array of vehicles that is either read in from a script, or initialized randomly. The file forward_observer.clp must be modified if you wish to change either of the following parameters:

- The observer's initial position, and the vehicle upon which he is mounted.
- The initial positions of the howitzers in the firing platoon.
- The types of vehicles that the observer is to recognize as hostile.

The location of the observer is in local X-Z coordinates. This vehicle will appear in the world along with the other vehicles that are read in from ./data/vehposfiles/ vehiclepos.cnvy, or the default random vehicles. The default observer location is the center of the world, as in:

```
(location observer 25000.0 25000.0)
```

The kind of vehicle that the observer is mounted upon is an index into the `vehtypearray` data structure declared in ./headers/npsnet.h. The association between indices and vehicles can be found in the file ./datafiles/vehicle.dat. The default is '8' for an M577, as in:

```
(mounted_on 8)
```

The observer's mission is specified in terms of vehicles to watch for, and what to do when he sees these vehicles. This means that you can program the observer for stupidity. The general template for these specifications is:

```
(mission <action> <vehicle-index>)
```

"<vehicle-index>"is an index to the vehtypearray, as mentioned above. "<action>" is either 'destroy', 'neutralize', or 'special'.

The default mission is just to kill T-72s, BMPs, and T2freds, as follows:

```
(mission destroy 51)
(mission destroy 52)
(mission destroy 53)
```

The location of each gun in a four-gun firing platoon defaults to the southeastern portion of the Fort Hunter-Ligget terrain database:

```
(location gun1 16000.0 16000.0)
(location gun2 25505.0 25506.0)
(location gun3 24000.0 24000.0)
(location gun4 24500.0 24500.0)
```

LIST OF REFERENCES

Atwood, D. J., "Modeling and Simulation Management Plan,", unpublished memorandum from then Deputy Secretary of Defense, dated 21 June 1991.

Fire Control Inputs for Cannon, 155mm Howitzer, M185 on Howitzer, Medium, Self-Propelled, 155mm, M109A1, Howitzer, Medium, Self-Propelled, 155mm, M109A1B, Howitzer, Medium, Self-Propelled, 155mm, M109A2, and Howitzer, Medium, Self-Propelled, 155mm, M109A3, and Cannon, 155mm Howitzer, M199 on Howitzer, Medium, Towed, 155mm, M198 Firing Projectile, HE, M107 With Fuzes, PD, M557, M572, and M739, Fuzes, MTSQ, M564, M520, M520A1, M500A1 and M582, Fuze, ET, M587, Fuzes, VT, M728, M514A3, M514, M514B1, M514A1 and M32, Fuzes, CP, M78 and M78A1, FCI 155-AM-B, Ballistic Research Laboratory, February 1983.

Branley, W. C. Jr., "Modeling Observation in Intelligent Agents: Knowledge and Belief," M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1992.

Culpepper, M. E., "Tactical Decision Making in Intelligent Agents: Developing Autonomous Forces in NPSNET," M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1992.

Charniak, and McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley Publishing Co., 1985.

Drummond, W. T. Jr., and Nizolak, J. P. Jr., "A Graphics Workstation Field Artillery Forward Observer Simulation Trainer," M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1989.

Firing Tables for Cannon, 155mm Howitzer, M185 on Howitzer, Medium, Self-Propelled, 155mm, M109A1, Howitzer, Medium, Self-Propelled, 155mm, M109A1B, Howitzer, Medium, Self-Propelled, 155mm, M109A2, and Howitzer, Medium, Self-Propelled, 155mm, M109A3, and Cannon, 155mm Howitzer, M199 on Howitzer, Medium, Towed, 155mm, M198 Firing Projectile, HE, M107; Projectile, Smoke, WP, M110; Projectile, Smoke, BE, M116, M116B1 (HC and Colored); Projectile, Smoke, BE, M116A1 (HC); Projectile, Gas, Persistent, H and HD, M110; Projectile, Gas, Nonpersistent, GB, M121A1; Projectile, Gas, Persistent, VX, M121A1; Projectile, Illuminating, M485A2 and M485A1, FT 155-AM-2, Department of the Army, March 1983.

JANUS(T) Documentation, Department of the Army, TRADOC Analysis Center, White Sands Missile Range, New Mexico, June 1986.

Garvey, R. E., Jr., and Monday, P., "SIMNET (SIMulator NETworking)," BBN Systems and Technologies, Bellevue, WA July 28, 1988.

How to Train With ARTBASS, Department of the Army, September 1987.

King, D M., and Prevatt, R. M. III, "Rapid Production of Graphical User Interfaces," M.S. Thesis, Naval Postgraduate School, Monterey, CA, December 1990.

Lieske, R. F., and Reiter, M. L., "Equations of Motion for A Modified Point Mass Trajectory," *Ballistics Research Laboratories Report Number 1314*, March 1966.

Loke, T., Tan, D., Seah, H., and Er, M., "Rendering Fireworks Displays," *IEEE Computer Graphics*, Vol. 12, No. 3, May 1992.

Maes, P., "Situated Agents Can Have Goals," *Designing Autonomous Agents*, Elsevier Science Publishers, 1990.

Monahan, J. G., "NPSNET: Physically-based Modeling Enhancements To An Object File Format," M. S. Thesis, Naval Postgraduate School, Monterey, CA, September 1991.

Osborne, W. D., "NPSNET: An Accurate Low-cost Technique for Real-time Display of Transient Events: Vehicle Collisions, Explosions and Terrain Modifications," M. S. Thesis, Naval Postgraduate School, Monterey, CA, September 1991.

Pylyshyn, A. W., *The Robot's Dilemma*, Able Publishing, 1987.

Pope, A., "The SIMNET Network and Protocols," *BBN Report No. 7102*, BBN Systems and Technologies, July 1989.

Reches, M., "Weapon Effectiveness and Casualty Reduction Analysis," *Ballistic Materials and Penetration Mechanics*, Elsevier Scientific Publishing Company, 1980.

Reeves, W. T., and Blau, R., "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems," *Proceedings of SIGGRAPH*, 1985.

Reeves, W. T., "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects," *acm Transactions on Graphics*, Vol. 2, No. 2, April 1983.

Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

Tanenbaum, A. S., *Structured Computer Organization*, Prentice Hall, 1990.

Walter, J. C., and Warren, P. T., "An Interactive, Three-Dimensional Application for Terrain Generation, Real Time Visualization, and Combat Scenario Reconstruction for the JANUS Combat Model," M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1992.

Wilson, K. P., "Naval Postgraduate School Class Library," unpublished user's manual for C++ container class software, July 1992.

Wilson, K. P., "NPSGDL: An Object Oriented Graphics Description Language for Virtual World Application Support," M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1992.

Zyda, M. J., "Workstation Parallel Programming," *Graphics and Video Laboratory Course Notes: Book 8*, unpublished text, 1991.

Zyda, M. J., McGhee, R.B., Ross, R., Smith, D., and Streyle, D., "Flight Simulators for Under $100,000," *IEEE Computer Graphics & Applications*, Vol. 8, No. 1, January 1988.

Zyda, M. J., and Pratt, D. R., "NPSNET: A 3D Visual Simulator for Virtual World Exploration and Experimentation," *1991 SID International Symposium Digest of Technical Papers*, Volume XXII, May 1991.

Zyda, M. J., Pratt, David R., Monahan, J. G., and Wilson, K. P., "NPSNET: Constructing A 3D Virtual World," *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, April 1992.

Zyda, M. J., Wilson, K. P., Pratt, D. R., and Monahan, J. G., "NPSOFF: An Object Description Language for Supporting Virtual World Construction," unpublished paper, October 1991.

BIBLIOGRAPHY

Brodie, K. W., *Mathematical Methods in Computer Graphics and Design*, Academic Press, 1980.

Burden, R. L., Faires, J. D., and Reynolds, A. C., *Numerical Analysis*, Prindle, Weber & Schmidt, 1981.

Chasen, S. H., *Geometric Principles and Procedures for computer Graphics Applications*, Prentice-Hall, 1978.

Department of the Army, *FM 101-61-2: Joint Munitions Effectiveness Manual, Surface to Surface Weapons, Ammunition and Fuze Characteristics*, October 1989.

Department of the Army, *AMCP 706-107: Elements of Armaments Engineering Part Two: Ballistics*, September 1963.

Department of the Army, *AMCP 706-140: Trajectories, Differential Effects, and Data for Projectiles*, August 1963.

Department of the Army, *AMCP 706-244: Research and Development of Material Engineering Design Handbook, Ammunition Series: Section 1, Artillery Ammunition, General*, September 1963.

Faiman, M., and Nievergelt, J., *Pertinent Concepts in Computer Graphics*, University of Illinois Press, 1969.

Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, 1990.

Gray, A., *Gyrostatics and Rotational Motion*, Macmillan Publishing Co., 1918.

Halliday, D., and Resnick, R., *Physics*, 3rd edition, John Wiley & Sons Inc., 1978.

Maron, M. J., *Numerical Analysis*, Macmillan Publishing Co., 1987.

Pavlidis, T., *Algorithms for Graphics and Image Processing*, Computer Science Press, 1982.

Rogers, D. F., and Adams, J. A., *Mathematical Elements for Computer Graphics*, McGraw-Hill, 1976.

U. S. Army Field Artillery School, *Field Artillery Cannon Weapon Systems and Ammunition, FC 6-50-19*, February 1984.

U. S. War Department, *A Course in Exterior Ballistics*, 1920.

Zill, D. G., *A First Course in Differential Equations*, Prindle, Weber, and Schmidt, 1979.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexanderia, VA 22304-6145	2
2.	Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dr. Michael J. Zyda Code CS/Zk Naval Postgraduate School Monterey, CA 93943-5000	4
4.	David R. Pratt Code CS/Pr Naval Postgraduate School Monterey, CA 93943-5000	4
5.	HQDA, OCSA, AI Center ATTN: CSDS-AI The Pentagon, Room 1D659 Washington, DC 20310-6900	1
6.	CPT David Nash 2448 Flora Avenue Fort Myers, FL 33907	1
7.	LCDR Don Brutzman Code OR/Br Naval Postgraduate School Monterey, CA 93943-5000	1